DOCUMENT RESUME

ED 243 168 CS 504 596

AUTHOR Sanford, David L.
TITLE The Application of Logic Programming to Communication Education.
PUB DATE Apr 84
NOTE 12p.; Paper presented at the Annual Meeting of the Southern Speech Communication Association (Baton Rouge, LA, April 5-7, 1984).
PUB TYPE Speeches/Conference Papers (150) -- Viewpoints (120) -- Information Analyses (070)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Communication Research; *Computer Assisted Instruction; *Computer Software; Educational Innovation; Higher Education; *Logic; *Speech Communication; Technological Advancement
IDENTIFIERS *Syllogisms

ABSTRACT

Recommending that communication students be required to learn to use computers not merely as number crunchers, word processors, data bases, and graphics generators, but also as logical inference makers, this paper examines the recently developed technology of logical programing in computer languages. It presents two syllogisms and shows how they would be represented in PROLOG, a logic-based computer language. It also suggests that the assumptions underlying some syllogisms must be directly stated for the computer to correctly interpret the data. (MM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED243168

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

David L. Sanford

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

THE APPLICATION OF LOGIC PROGRAMMING TO COMMUNICATION EDUCATION

David L. Sanford
Dept. of Communication Studies

Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Running Title: Logic Programming

Copyright © 1984 David L. Sanford

Paper presented to the panel on Computers in Communication at the Southern Speech Communication Association Convention in Baton Rouge, LA, April, 1984.

CS504596

> It is conventional to speak of computer *languages* in reference to the instructions fed into the machine: but, strictly speaking, these are not languages but rather *codes*. The various functional operations carried out by a computer (multiplying, adding, etc.) are certainly operational and therefore semantic; nevertheless, they are *definable*, and remain at the syntactic level. It is the human programmer who carries out the cognitive, semantic processes and converts them into syntactic instructions. There was a fashion some 25 years ago, but now fortunately passing, to compare computers with brains. It was a revival of a concern in the seventeenth and eighteenth centuries with robots, perhaps stemming from the work of Leibnitz and Decartes upon "logic machines." However, logical reasoning operates, not with ordinary language, but with *language-systems* that use a fixed range of signs and rules of usage defined in a metalanguage. ... There is no particular cultural merit in numeration, as such, nor (unless you are a designer or repairman) in understanding the internal operations of any machine. (Cherry, 1980, pp. 262-263, italics in original)

In the essay quoted above, Colin Cherry seems to be saying that computers are fine calculating machines for mathematical computations, but that the logic of cognitive processes is different in some fundamental way. Indeed, he and others seem to feel that the notion of computers as "logic machines" is incorrect: that computers can never be more than rapid and complex adding machines. Lately, it has become increasingly clear that this notion of the inherent limitation of computers is wrong, that they indeed are logic machines.

This paper introduces logic programming computer languages as potentially important to communication education. We suggest that communication students should be required to learn computer programming, particularly logic programming. Not only will all students need to be prepared to understand and use computers in the work environment, but communication students have traditionally been required to study logic as a component of rhetoric and/or forensics. The recent

development of logic programming uniquely combines these goals in a way that many students may find challenging, rewarding, and even pleasurable.

Computers traditionally have been applied to communication education in at least four ways. First, computers were originally considered as complex and versatile adding machines. Therefore, students of communication were expected to learn to use computers as "number-crunchers", i.e., as mechanisms for conducting the statistical analyses needed in communication research. Second, computers rapidly are replacing typewriters as mechanisms for producing clean, readable papers. Students in all disciplines are being expected to produce neater, more polished term papers thanks to the control over formatting and error correction afforded by word-processing. Third, computerized data-bases, such as the *New York Times Information Retrieval* system, allow students to do extensive library research without having to expend many hours of manual labor with a card-catalog. And fourth, but probably least frequently to date, students of visual communication are being taught the principles of visual and graphic design by learning to program computer graphics.

In the Department of Communication Studies at this University, we have being using all 4 of the above functions of computers in education for the last several years. A few of our best students take a senior-level sequence on Communication Theory and Research, where they write, conduct, and analyze their own surveys, requiring that they be

introduced to using statistical packages like SPSS (Nie et al., 1975). All students in that sequence, as well as other upperclass sequences in communication, are required to use the computer's word-processing capabilities to produce their term papers. The University has the *Dialog* system available on campus, to allow one to conduct computerized literature searches. Several of our students use the system, although it is not required in any of our courses. And finally, just recently, this author introduced an independent study on Modern Technologies for Visual Communication in which we taught computer graphics programming.

These 4 functions are only the beginning for computer uses in communication education. The basic nature of computer technology is undergoing a dramatic change in form and function that will have a tremendous impact. Computers are no longer considered to be just rapid and complex calculators, but are now appreciated as symbolic manipulators that make logical inferences. In times past when teaching rhetoric or forensics, we encouraged students to study *logos* by taking a logic course from the Philosophy department. Due to the development of logic programming on computers, we should begin requiring students to learn to program in these logic-based computer languages in order to understand both *logos* and computers.

Logic Programming

Logic programming was first proposed by Post (1943), but no implementation was available until the early 1970's, when PROLOG (for PROgramming in LOGic) was developed in France (Kowalski, 1979). Currently, there are several different versions available, each with a slightly different syntax, different built-in functions or rules, and different names. Other implementations include Q-LOG (Komorowski, 1982), LOGLISP (Robinson & Sibert, 1980), HCPRVR (Chester, 1979), and HC (Roach & Fowler, 1982). For a fuller discussion of the similarities and differences between logic programming and traditional computer languages like FORTRAN and BASIC, see Sanford and Roach (1984).

In order to appreciate how PROLOG is usable directly to teach and program logic as used in communication education, the programming of a couple of syllogisms will be presented. First, a discussion of alternative syntactic forms for the expression of logical relationships is presented. Then, transcripts of the interactive operation of a computer PROLOG interpreter and a human user exploring the logic of 2 syllogisms is discussed.

The first syllogism is the classic example that follows:

All men are mortal.

Socrates is a man.

This is translated into standard first-order predicate logic form as the following:

$$\forall x\ (M_a(x) \rightarrow M_o(x))$$

$$\exists x\ (N(x, \text{Socrates})\ \&\ M_a(x))$$

The first statement is read: for all x it is true that x is a member of the class of Man implies that x is a member of the class of Mortal. The second statement is read: there exists an x for which it is true that the name of x is Socrates and that x is a member of the class of Man. Certainly, here is the metalanguage of logic discussed by Colin Cherry.

PROLOG is another form of a first-order predicate logic, termed Horn clauses (Horn, 1951). While retaining the mathematical precision of traditional first-order predicate logic, it also comes closer to natural language structures. The above statements of the major and minor premises of a syllogism would look like the following in PROLOG:

```
is-a(?x mortal) if is-a(?x man).
is-a(Socrates man).
```

Since this is a computer programming language, it must distinguish between constants and variables. In this version, any word that begins with a question mark is understood to be a variable. Therefore, the first statement above would be read as: x is a mortal if x is a man. The second statement above similarly would be read as: Socrates is a man.

At this point, one has sufficient background about the syntax of PROLOG to follow the interaction between a human programmer and a computer PROLOG interpreter. A final note about the transcript below is needed. The percent sign (%) is a user prompt, i.e., it appears in front of what the programmer types into the computer (the input). The

computer's responses (output) appear on the lines with no prompt.

In order for the computer to remember a fact, one must explicitly tell the computer to remember it, by asserting it, as follows:

```
% assert( is-a(?x mortal) if is-a(?x man). ).
ok
% assert( is-a(Socrates man). ).
ok
```

This stores these facts or rules in working memory. Now the user can input queries and expect useful output.

```
% is-a(Socrates ?x).
```

In other words, the user is asking: Socrates is a what?

```
is-a(Socrates mortal).
```

The response is that Socrates is a mortal.

```
% is-a(?x mortal).
```

Here, the programmer is asking: who (or what) is a mortal?

```
is-a(Socrates mortal).
```

Again, the response is that Socrates is a mortal. Note that although this was logically calculated for the first query it was not asserted, therefore after outputing it to the user the computer immediately forgot it. Consequently when the second question was posed, the computer had to logically recompute the same answer. Also note that it is not necessary to ask a question in this manner. The following is a more specific question and its subsequent answer.

```
% is-a(Socrates mortal).
is-a(Socrates mortal).
```

What if the PROLOG interpreter is asked a question it cannot answer? Here is an example of asking for information the computer does not know:

```
% is-a(Socrates woman).
false
```

Although the response is "false", that does not mean the statement it truly "false" in a general sense. Take the following example:

```
% is-a(Socrates Greek).
false
```

This simply means that the computer can not prove the statement with the facts it currently has available.

Now examine another syllogism. The English version of these premises are:

All dogs are canines.

All terriers are dogs.

First one asserts the premises. Then one asks a series of questions.

```
% assert( is-a(?x canine) if is-a(?x dog). ).
ok
% assert( is-a(?x dog) if is-a(?x terrier). ).
ok
% is-a(terrier ?x).
false
% is-a(dog ?x).
false
% is-a(canine ?x).
false
```

```
% is-a(?x canine).
false
% is-a(?x dog).
false
% is-a(?x terrier).
false
```

Why is the computer having trouble here? The conclusion of the syllogism is obvious: all terriers are canines. This is part of the true value of exploring logic with a computer. What is obvious to a human being is not obvious to a computer. Clearly, as far as the computer is concerned, some vital fact is missing from the set of rules stored in its memory. This fact is not missing from the memory of humans, since we have no trouble solving the syllogism. But it may not be clear what that missing fact is. Indeed, Hopper (1981) argues that people always leave out details that other people usually have no trouble filling in. He indicates, for example, that the missing premise of an enthymeme is a taken-for-granted. This example seems to indicate that even a syllogism has something taken-for-granted that makes it solvable. One should take their lead from Aristotle, who was astute enough to avoid taking some things for granted. By asserting the following additional rule or fact, the syllogism is finally solvable:

```
% assert( is-a(?x ?x). ).
ok
% is-a(terrier ?x).
is-a(terrier canine).
```

Aristotle presented as one of the basic tenets of logic that all A is A.

This would lend support to Toulmin's (1958) argument that the datum, warrant, and claim are not all of the major parts of a logical argument; it requires backing, as well.

Conclusion

This paper examined the recently developed technology of logic programming computer languages. It presented 2 syllogisms and showed how they would be represented in PROLOG and how they would be solved by such a logic interpreter. The intention was to introduce the issue of using computers as "logic machines". That is why this paper stops short of moving beyond the Aristotlian syllogism to the Toulmin model of argument. Horn clauses can be used to represent a Toulmin model of argument just as readily as they were used above for syllogisms. Also, Horn clauses can be written that would take the given statements of an enthymeme and discover the missing premise, although that is not nearly as direct or easy as solving a syllogism.

The main goal of this paper, however, was to suggest that computers can and should have a dramatic impact on communication education. Students of communication need to learn to use computers not merely as number-crunchers, word-processors, data-bases, and graphics-generators, but also as logical inference-makers.

References

Cherry, C. The communication explosion. In M. L. Foster & S. H. Brandes (Eds.) *Symbol as sense: New approaches to the analysis of meaning*. New York: Academic Press, 1980, 249-267.

Chester, D. L. Using HCPRVR (Tech. Rep.). Austin, TX: U. of Texas, Dept. of Computer Science, 1979.

Hopper, R. The taken-for-granted. *Human Communication Research*, 1981, *7*, 195-211.

Horn, A. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 1951, *16*, 14-21.

Komorowski, H. J. QLOG--The programming environment for PROLOG in LISP. In K. L. Clark & S.-A. Tarnlund (Eds.), *Logic Programming*. New York: Academic Press, 1982, 315-322.

Kowalski, R. A. *Logic for problem solving*. New York: North Holland, 1979.

Nie, N. H., Hull, C. H., Jenkins, J. G., Steinbrenner, K., & Bent, D. H. *SPSS: Statistical package for the social sciences* (2nd ed.). New York: McGraw-Hill, 1975.

Post, E. L. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 1943, *65*, 197-215.

Roach, J. W. & Fowler, G. S. The HC manual: Virginia Tech PROLOG (Tech. Rep.). Blacksburg, VA: Virginia Tech, Dept. of Computer Science, 1982.

Robinson, J. A. & Sibert, E. E. LOGLISP--An alternative to PROLOG (Tech. Rep.). Syracuse, NY: Syracuse U., School of Information and Computer Science, 1980.

Sanford, D. L. & Roach, J. W. The application of artificial intelligence as a methodology for human communication research. Paper presented to the Southern Speech Communication Association Convention at Baton Rouge, LA, April, 1984.

Toulmin, S. E. *The uses of argument*. Cambridge, England: Cambridge University Press, 1958.